

FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111111111	111111111	AAA	AAA
FFF	111111111	111111111	AAA	AAA
FFF	111111111	111111111	AAA	AAA

CHI

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIIIIi    SSSSSSSS
LLLLLLLLLLLL IIIIIIII    SSSSSSSS

```



```
1 0001 0 MODULE CHKPRO (  
2 0002 0 LANGUAGE (BLISS32),  
3 0003 0 IDENT = 'V04-000'  
4 0004 0 ) =  
5 0005 1 BEGIN  
6 0006 1  
7 0007 1  
8 0008 1 *****  
9 0009 1 *  
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *  
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *  
12 0012 1 * ALL RIGHTS RESERVED. *  
13 0013 1 *  
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *  
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *  
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *  
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *  
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *  
19 0019 1 * TRANSFERRED. *  
20 0020 1 *  
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *  
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *  
23 0023 1 * CORPORATION. *  
24 0024 1 *  
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *  
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *  
27 0027 1 *  
28 0028 1 *  
29 0029 1 *****  
30 0030 1  
31 0031 1 ++  
32 0032 1  
33 0033 1 FACILITY: F11ACP Structure Level 1  
34 0034 1  
35 0035 1 ABSTRACT:  
36 0036 1  
37 0037 1 This routine checks the volume and file protection to see if the  
38 0038 1 user is authorized to perform the intended operation.  
39 0039 1  
40 0040 1 ENVIRONMENT:  
41 0041 1  
42 0042 1 STARLET operating system, including privileged system services  
43 0043 1 and internal exec routines.  
44 0044 1  
45 0045 1 --  
46 0046 1  
47 0047 1  
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 2-May-1977 10:45  
49 0049 1  
50 0050 1 MODIFIED BY:  
51 0051 1  
52 0052 1 V03-002 LMP0221 L. Mark Pilant, 29-Mar-1984 6:50  
53 0053 1 Change UCB$$_OWNUIC to ORB$$_OWNER and UCB$$_VPROT to  
54 0054 1 ORB$$_PROT. Also, change over to the new CHKPRO interface.  
55 0055 1  
56 0056 1  
57 0057 1 V03-001 LMP0154 L. Mark Pilant, 13-Sep-1983 15:29
```

CHKPRO
V04-000

E 8
16-Sep-1984 00:49:44 VAX-11 Bliss-32 V4.0-742 Page 2
14-Sep-1984 12:29:21 DISK\$VMMASTER:[F11A.SRC]CHKPRO.B32;1 (1)

```

: 58      0058 1 |      Re-write to use the common protection checking routine.
: 59      0059 1 |
: 60      0060 1 |      A0102  ACG0089      Andrew C. Goldstein,      6-Dec-1979  17:43
: 61      0061 1 |      Fix uninitialized pointer
: 62      0062 1 |
: 63      0063 1 |      A0101  ACG0047      Andrew C. Goldstein,      8-Aug-1979  22:02
: 64      0064 1 |      Add SYSPRV privilege, protection interface changes
: 65      0065 1 |
: 66      0066 1 |      A0100  ACG00001      Andrew C. Goldstein, 10-Oct-1978  20:02
: 67      0067 1 |      Previous revision history moved to F11A.REV
: 68      0068 1 |      **
: 69      0069 1 |
: 70      0070 1 |
: 71      0071 1 |      LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 72      0072 1 |      REQUIRE 'SRC$:FCPDEF.B32';
: 73      0387 1 |
: 74      0388 1 |      FORWARD ROUTINE
: 75      0389 1 |      CHECK_PROTECT,
: 76      0390 1 |      CHECK_PROT;
```



```

: 78      0391 1 GLOBAL ROUTINE CHECK_PROTECT (ACCESS, HEADER, FCB, ALT_ACCESS, REQUIRED) =
: 79      0392 1
: 80      0393 2 BEGIN
: 81      0394 2 LOCAL STATUS;
: 82      0395 3 STATUS = (IF ACTUALCOUNT GEQU 5
: 83      0396 4 THEN KERNEL_CALL (CHECK_PROT, .ACCESS, .HEADER, .FCB, .ALT_ACCESS, .REQUIRED)
: 84      0397 4 ELSE KERNEL_CALL (CHECK_PROT, .ACCESS, .HEADER, .FCB, 0, 0)
: 85      0398 2 );
: 86      0399 2 IF NOT .STATUS THEN ERR_EXIT (.STATUS) ELSE RETURN .STATUS;
: 87      0400 1 END;
```

```
.TITLE CHKPRO
.IDENT \V04-000\
```

```
.EXTRN SYS$CMKRNL
```

```
.PSECT $CODE$,NOWRT,2
```

			0000	00000		.ENTRY	CHECK_PROTECT, Save nothing		0391
	05	6C	91	00002		CMPB	(AP), #5		0395
		06	1F	00005		BLSSU	1\$		
	7E	10	AC	7D 00007		MOVQ	ALT_ACCESS, -(SP)		0396
		02	11	0000B		BRB	2\$		
		7E	7C	0000D	1\$:	CLRQ	-(SP)		0397
	7E	08	AC	7D 0000F	2\$:	MOVQ	HEADER, -(SP)		
		04	AC	DD 00013		PUSHL	ACCESS		
			05	DD 00016		PUSHL	#5		
			5E	DD 00018		PUSHL	SP		
		0000V	CF	9F 0001A		PUSHAB	CHECK_PROT		
00000000G	9F		08	FB 0001E		CALLS	#8, @SYS\$CMKRNL		
	02		50	EB 00025		BLBS	STATUS, 3\$		0399
			50	BF 00028		CHMU	STATUS		
			04	0002A	3\$:	RET			0400

```
; Routine Size: 43 bytes, Routine Base: $CODE$ + 0000
```

```

: 88      0401 1
: 89      0402 1 ROUTINE CHECK_PROT (ACCESS, HEADER, FCB, ALT_ACCESS, REQUIRED) =
: 90      0403 1
: 91      0404 1 ++
: 92      0405 1
: 93      0406 1 FUNCTIONAL DESCRIPTION:
: 94      0407 1
: 95      0408 1 This routine checks the volume and file protection to see if the
: 96      0409 1 user is authorized to perform the intended operation.
: 97      0410 1
: 98      0411 1 CALLING SEQUENCE:
: 99      0412 1 , CHECK_PROTECTION (ARG1, ARG2, ARG3, ARG4, ARG5, ARG6)
: 100     0413 1
: 101     0414 1 INPUT PARAMETERS:
: 102     0415 1 ARG1: access mode
: 103     0416 1 READ_ACCESS = 0
: 104     0417 1 WRITE_ACCESS = 1
: 105     0418 1 DELETE_ACCESS = 2
: 106     0419 1 CREATE_ACCESS = 3
```

```
107 0420 1 | RDATT_ACCESS = 4
108 0421 1 | WRATT_ACCESS = 5
109 0422 1 | ARG2: address of file header, if any
110 0423 1 | ARG3: address of FCB or 0
111 0424 1 | ARG4: (optional) alternate access mask to test for
112 0425 1 | ARG5: (optional) 1 if alternate access if required
113 0426 1 | 0 if optional
114 0427 1 |
115 0428 1 | IMPLICIT INPUTS:
116 0429 1 | CURRENT_UCB: address of device UCB
117 0430 1 | IO_PACKET: I/O packet of this request
118 0431 1 |
119 0432 1 | OUTPUT PARAMETERS:
120 0433 1 | NONE
121 0434 1 |
122 0435 1 | IMPLICIT OUTPUTS:
123 0436 1 | NONE
124 0437 1 |
125 0438 1 | ROUTINE VALUE:
126 0439 1 | NONE
127 0440 1 |
128 0441 1 | SIDE EFFECTS:
129 0442 1 | NONE
130 0443 1 |
131 0444 1 | --
132 0445 1 |
133 0446 2 | BEGIN
134 0447 2 |
135 0448 2 | MAP
136 0449 2 | HEADER : REF BBLOCK, ! File header arg
137 0450 2 | FCB : REF BBLOCK; ! FCB arg
138 0451 2 |
139 0452 2 | LINKAGE
140 0453 2 | L_CHKPRO_INT = JSB (REGISTER = 0, REGISTER = 1,
141 0454 2 | REGISTER = 2, REGISTER = 3);
142 0455 2 |
143 0456 2 | LABEL
144 0457 2 | CHECK_BLOCK; ! body of a single check attempt
145 0458 2 |
146 0459 2 | LOCAL
147 0460 2 | STATUS, ! Local routine exit status
148 0461 2 | FILE_ACCESS_BITS: BBLOCK [1], ! Actual access mask to file
149 0462 2 | PROTECTION, ! Protection code of file
150 0463 2 | OWNER_UIC, ! File owner UIC
151 0464 2 | SEG_NUMBER, ! Segment number of file header
152 0465 2 | CHPCTL : BBLOCK [CHPCTL$C_LENGTH], ! CHKPRO control block
153 0466 2 | CHPRET : BBLOCK [CHPRET$C_LENGTH], ! CHKPRO return arg block
154 0467 2 | ARB : REF BBLOCK, ! Agent's rights block
155 0468 2 | ORB : REF BBLOCK, ! Object's rights block
156 0469 2 | LOCAL_ORB : BBLOCK [ORB$C_LENGTH], ! Needed when no FCB given
157 0470 2 | MAP_AREA : REF BBLOCK, ! Address of header map area
158 0471 2 | PRIVS_USED : BBLOCK [4]; ! Privileges used
159 0472 2 |
160 0473 2 | BIND ! Access mode tables
161 0474 2 | ! Write operation on volume
162 0475 2 | WRITE_OP = UPLIT (
163 0476 2 | ARMSM_WRITE OR ARMSM_DELETE OR ARMSM_CONTROL),
```



```
164 0477 2
165 0478
166 0479      ! no READALL privilege for operation
167 0480      = UPLIT (
168 0481      ARMSM_WRITE OR ARMSM_DELETE),
169 0482
170 0483      ! Check for zero file segment number
171 0484      EXT_HEADER = UPLIT BYTE (
172 0485      %B'100111'
173 0486      ) : BITVECTOR,
174 0487
175 0488      ! Volume access bits
176 0489      VOL_ACCESS = UPLIT BYTE (
177 0490      ARMSM_READ,
178 0491      ARMSM_READ OR ARMSM_WRITE,
179 0492      ARMSM_READ OR ARMSM_DELETE,
180 0493      ARMSM_READ OR ARMSM_WRITE,
181 0494      ARMSM_READ,
182 0495      ARMSM_READ OR ARMSM_WRITE
183 0496      ) : VECTOR [,BYTE],
184 0497
185 0498      ! File access bits
186 0499      FILE_ACCESS = UPLIT BYTE (
187 0500      ARMSM_READ,
188 0501      ARMSM_READ OR ARMSM_WRITE,
189 0502      ARMSM_DELETE,
190 0503      ARMSM_WRITE,
191 0504      ARMSM_READ,
192 0505      ARMSM_CONTROL
193 0506      ) : VECTOR [,BYTE];
194 0507
195 0508      EXTERNAL
196 0509      CLEANUP_FLAGS : BITVECTOR,      ! Cleanup action and status flags
197 0510      CURRENT_UCB   : REF BBLOCK,     ! Device UCB
198 0511      PRIMARY_FCB   : REF BBLOCK,     ! Primary FCB address
199 0512      IO_PACKET     : REF BBLOCK,     ! I/O packet of this request
200 0513      LOCAL_ARB     : BBLOCK;         ! local copy of caller's ARB
201 0514
202 0515      EXTERNAL ROUTINE
203 0516      ALLOCATE,      ! Get a block of non-paged pool
204 0517      DEALLOCATE,    ! Release a block of non-paged pool
205 0518      EX$CHKPRO_INT : L_CHKPRO_INT ADDRESSING_MODE (GENERAL);
206 0519      ! General purpose protection checker
207 0520
208 0521      ! Initialize the CHKPRO control block and the return arg block.
209 0522
210 0523      CH$FILL (0, CHPCTL$C_LENGTH, CHPCTL);
211 0524      CH$FILL (0, CHPRET$C_LENGTH, CHPRET);
212 0525      CHPRET[CHPRET$C_PRIVS_USED] = PRIVS_USED;
213 0526
214 0527      ! Derive the composite file access mask from the access type and
215 0528      ! the alternate access mask.
216 0529
217 0530
218 0531      FILE_ACCESS_BITS = .FILE_ACCESS[.ACCESS] OR .ALT_ACCESS;
219 0532
220 0533      ! We try the whole operation twice: once with the added alternate access
```

```
221 0534 2 ! mask, and if that fails, once without.
222 0535 2 !
223 0536 2
224 0537 2 WHILE 1 DO
225 0538 3 BEGIN
226 0539 4 CHECK_BLOCK: BEGIN ! scope of one try
227 0540 4
228 0541 4 ! If the requested operation is a write operation, check to make
229 0542 4 ! sure that the volume is not software write locked.
230 0543 4
231 0544 4 IF (.WRITE_OP AND .FILE_ACCESS_BITS) NEQ 0
232 0545 4 AND .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_SWL]
233 0546 4 THEN
234 0547 5 BEGIN
235 0548 5 STATUS = SS$WRITLCK;
236 0549 5 LEAVE CHECK_BLOCK;
237 0550 4 END;
238 0551 4
239 0552 4 ! Get the address of the Agent's Rights Block (ARB) and Object's Rights Block
240 0553 4 ! (ORB).
241 0554 4
242 0555 4 ARB = .IO_PACKET[IRP$L_ARB];
243 0556 4 ORB = .CURRENT_UCB[UCB$L_ORB];
244 0557 4
245 0558 4 ! Now check the volume protection to make sure that the requested operation
246 0559 4 ! is allowed. If the attempted access is denied, return with the error.
247 0560 4
248 0561 4 CHPCTL[CHPCTL$L_ACCESS] = .VOL_ACCESS[.ACCESS];
249 0562 4 IF .FILE_ACCESS_BITS[ARMSV_WRITE]
250 0563 4 OR .FILE_ACCESS_BITS[ARMSV_CONTROL]
251 0564 4 THEN BBLOCK [CHPCTL[CHPCTL$L_ACCESS], ARMSV_WRITE] = 1;
252 0565 4 IF .FILE_ACCESS_BITS[ARMSV_DELETE]
253 0566 4 THEN BBLOCK [CHPCTL[CHPCTL$L_ACCESS], ARMSV_DELETE] = 1;
254 0567 4 BBLOCK [CHPCTL[CHPCTL$L_FLAGS], CHP$V_READ] = 1;
255 0568 4 IF (.WRITE_OP AND .FILE_ACCESS_BITS) NEQ 0
256 0569 4 THEN BBLOCK [CHPCTL[CHPCTL$L_FLAGS], CHP$V_WRITE] = 1;
257 0570 4 IF (.NOREADALL AND .FILE_ACCESS_BITS) EQL 0
258 0571 4 THEN BBLOCK [CHPCTL[CHPCTL$L_FLAGS], CHP$V_USEREADALL] = 1;
259 0572 4
260 0573 4 ! Items to return
261 0574 4
262 0575 4 PRIVS_USED = 0;
263 0576 4
264 0577 4 STATUS = EX$CHKPRO_INT (.ARB, .ORB, CHPCTL, 0);
265 0578 4 IF NOT .STATUS
266 0579 4 THEN LEAVE CHECK_BLOCK;
267 0580 4
268 0581 4 ! If there is no FCB specified and no header given, it is a volume access
269 0582 4 ! check. In which case, control may be returned now.
270 0583 4
271 0584 4 IF .FCB EQL 0 AND .HEADER EQL 0
272 0585 4 THEN LEAVE CHECK_BLOCK;
273 0586 4
274 0587 4 ! Get the protection, owner, and segment number for the desired header.
275 0588 4
276 0589 4 IF .FCB NEQ 0
277 0590 4 THEN
```



```
278 0591 5 BEGIN
279 0592 5 ORB = FCB[FCB$R_ORB];
280 0593 5
281 0594 5 ! Note that control access is always denied for group and world access.
282 0595 5
283 0596 5 LOCAL_ORB[ORB$$_GRP_PROT] = ORB[ORB$$_GRP_PROT];
284 0597 5 LOCAL_ORB[ORB$$_WOR_PROT] = ORB[ORB$$_WOR_PROT];
285 0598 5 BBLOCK [ORB[ORB$$_GRP_PROT], ARMSV_CONTROL] = 1;
286 0599 5 BBLOCK [ORB[ORB$$_WOR_PROT], ARMSV_CONTROL] = 1;
287 0600 5 SEG_NUMBER = .FCB[FCB$$_SEGN];
288 0601 5 END
289 0602 4 ELSE
290 0603 5 BEGIN
291 0604 5 CH$FILL (0, ORB$$_LENGTH, LOCAL_ORB);
292 0605 5 ORB = LOCAL_ORB;
293 0606 5 ORB[ORB$$_SIZE] = ORB$$_LENGTH;
294 0607 5 ORB[ORB$$_TYPE] = DYN$$_ORB;
295 0608 5 ORB[ORB$$_PROT_16] = 1;
296 0609 5 ORB[ORB$$_PROT] = .HEADER[FH1$$_FILEPROT];
297 0610 5 ORB[ORB$$_UICMEMBER] = .HEADER[FH1$$_UICMEMBER];
298 0611 5 ORB[ORB$$_UICGROUP] = .HEADER[FH1$$_UICGROUP];
299 0612 5 MAP_AREA = .HEADER + .HEADER[FH1$$_MPOFFSET]*2;
300 0613 5 SEG_NUMBER = .MAP_AREA[FM1$$_EX_SEGNUM];
301 0614 4 END;
302 0615 4
303 0616 4 ! Next, if the operation is on an extension header, make sure that only the
304 0617 4 ! system is allowed access for most operations.
305 0618 4
306 0619 4 IF .EXT_HEADER[.ACCESS]
307 0620 4 THEN
308 0621 5 BEGIN
309 0622 5 IF .SEG_NUMBER GTR 0 AND NOT .CLEANUP_FLAGS[CLF_SYSPRV]
310 0623 5 THEN
311 0624 6 BEGIN
312 0625 6 STATUS = SS$$_NOPRIV;
313 0626 6 LEAVE CHECK_BLOCK;
314 0627 5 END;
315 0628 4 END;
316 0629 4
317 0630 4 ! Now check the access requested to determine if access is to be granted or
318 0631 4 ! denied.
319 0632 4
320 0633 4 CHPCTL[CHPCTL$$_ACCESS] = .FILE_ACCESS_BITS;
321 0634 4 PRIVS_USED = 0;
322 0635 4
323 0636 4 STATUS = EXE$CHKPRO_INT (.ARB, .ORB, CHPCTL, CHPRET);
324 0637 4
325 0638 4 ! Control implies read attribute access. The protection check needs
326 0639 4 ! to be retried in this case.
327 0640 4 !
328 0641 4
329 0642 4 IF NOT .STATUS
330 0643 4 THEN
331 0644 5 BEGIN
332 0645 5 IF .ACCESS EQL RDATT_ACCESS
333 0646 5 THEN
334 0647 6 BEGIN
```

```

335 0648 6      BBLOCK [CHPCTL[CHPCTL$ACCESS], ARMSV_READ] = 0;
336 0649 6      BBLOCK [CHPCTL[CHPCTL$ACCESS], ARMSV_CONTROL] = 1;
337 0650 6      PRIVS_USED = 0;
338 0651 6      STATUS = EXESCHKPRO_INT (.ARB, .ORB, CHPCTL, CHPRET);
339 0652 5      END;
340 0653 4      END;
341 0654 4
342 0655 4      ! If we just tried a protection check with alternate access and it
343 0656 4      ! failed, retry it with just the normal access. Otherwise, we are
344 0657 4      ! done.
345 0658 4
346 0659 4
347 0660 4      END;
348 0661 4      ! end of block CHECK_BLOCK
349 0662 4
350 0663 4      IF .STATUS
351 0664 4      OR .REQUIRED
352 0665 4      OR .FILE_ACCESS_BITS EQL .FILE_ACCESS[.ACCESS]
353 0666 4      THEN EXITLOOP;
354 0667 4
355 0668 4      FILE_ACCESS_BITS = .FILE_ACCESS[.ACCESS];
356 0669 4      END;
357 0670 4      ! end of retry loop
358 0671 4
359 0672 4      ! Check if the alternate access check failed. If so, return alternate
360 0673 4      ! success status.
361 0674 4
362 0675 4      IF .STATUS
363 0676 4      AND .ALT_ACCESS NEQ 0
364 0677 4      AND .FILE_ACCESS_BITS EQL .FILE_ACCESS[.ACCESS]
365 0678 4      THEN STATUS = SS$_NOTALLPRIV;
366 0679 4
367 0680 4      RETURN .STATUS
368 0681 1      END;
369 0682 1      ! End of routine CHECK_PROTECT
```

```

0000001A 0002B .BLKB 1
0000000A 0002C P.AAA: .LONG 26
0000000A 00030 P.AAB: .LONG 10
00000027 00034 P.AAC: .BYTE 39
03 01 03 09 03 01 00035 P.AAD: .BYTE 1, 3, 9, 3, 1, 3
10 01 02 08 03 01 0003B P.AAE: .BYTE 1, 3, 8, 2, 1, 16

WRITE_OP= P.AAA
NOREADALL= P.AAB
EXT_HEADER= P.AAC
VOL_ACCESS= P.AAD
FILE_ACCESS= P.AAE
.EXTRN CLEANUP_FLAGS, CURRENT_UCB
.EXTRN PRIMARY_FCB, IO_PACKET
.EXTRN LOCAL_ARB, ALLOCATE
.EXTRN DEALLOCATE, EXESCHKPRO_INT

OFFC 0000 CHECK_PROT:
5E FF6C CE 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
MOVAB -148(SP), SP ; 0402
```


0C	00	6E		00	2C	00007	MOV C5	#0, (SP), #0, #12, CHPCTL	0523
28	00	6E	F4	AD	2C	0000C	MOV C5	#0, (SP), #0, #40, CHPRET	0524
			60	AE		00013			
	F0	AD	04	AE	9E	00015	MOVAB	PRIVS_USED, CHPRET+36	0525
		50	DD	AF	9E	0001A	MOVAB	FILE_ACCESS, R0	0531
59		50	04	AC	C1	0001E	ADDL3	ACCESS, R0, R9	
57		69	10	AC	89	00023	BISB3	ALT_ACCESS, (R9), FILE_ACCESS_BITS	
				51	D4	00028	1\$: CLRL	R1	0544
		57	BE	AF	93	0002A	BITB	WRITE_OP, FILE_ACCESS_BITS	
				14	13	0002E	BEQL	3\$	
				51	D6	00030	INCL	R1	
	08	50	0000G	CF	D0	00032	MOVL	CURRENT_UCB, R0	0545
		A0		01	E1	00037	BBC	#1, 59(R0), 3\$	
		58	025C	8F	3C	0003C	MOVZWL	#604, STATUS	0548
				0124	31	00041	2\$: BRW	14\$	0549
		50	0000G	CF	D0	00044	3\$: MOVL	IO_PACKET, R0	0555
		5A	58	A0	D0	00049	MOVL	88(R0), ARB	
		50	0000G	CF	D0	0004D	MOVL	CURRENT_UCB, R0	0556
		56	1C	A0	D0	00052	MOVL	28(R0), ORB	
		50	9B	AF	9E	00056	MOVAB	VOL_ACCESS, R0	0561
	F4	AD	04	BC40	9A	0005A	MOVZBL	@ACCESS[R0], CHPCTL	
04		57		01	E0	00060	BBS	#1, FILE_ACCESS_BITS, 4\$	0562
04		57		04	E1	00064	BBC	#4, FILE_ACCESS_BITS, 5\$	0563
	F4	AD		02	88	00068	4\$: BISB2	#2, CHPCTL	0564
04		57		03	E1	0006C	5\$: BBC	#3, FILE_ACCESS_BITS, 6\$	0565
	F4	AD		08	88	00070	BISB2	#8, CHPCTL	0566
	F8	AD		01	88	00074	6\$: BISB2	#1, CHPCTL+4	0567
		04		51	E9	00078	BLBC	R1, 7\$	0568
	F8	AD		02	88	0007B	BISB2	#2, CHPCTL+4	0569
		57	FF6C	CF	93	0007F	7\$: BITB	NOREADALL, FILE_ACCESS_BITS	0570
				04	12	00084	BNEQ	8\$	
	F8	AD		04	88	00086	BISB2	#4, CHPCTL+4	0571
		52	04	AE	D4	0008A	8\$: CLRL	PRIVS_USED	0575
			F4	AD	9E	0008D	MOVAB	CHPCTL, R2	0577
				53	D4	00091	CLRL	R3	
		51		56	D0	00093	MOVL	ORB, R1	
		50		5A	D0	00096	MOVL	ARB, R0	
			00000000G	00	16	00099	JSB	EXE\$CHKPRO_INT	
		58		50	D0	0009F	MOVL	R0, STATUS	
		03		58	E8	000A2	BLBS	STATUS, 9\$	0578
				00C3	31	000A5	BRW	15\$	
		51	0C	AC	D0	000A8	9\$: MOVL	FCB, R1	0584
				05	12	000AC	BNEQ	10\$	
			08	AC	D5	000AE	TSTL	HEADER	
				8E	13	000B1	BEQL	2\$	
				51	D5	000B3	10\$: TSTL	R1	0589
				1C	13	000B5	BEQL	11\$	
		56	58	A1	9E	000B7	MOVAB	88(R1), ORB	0592
28		AE	20	A6	9E	000BB	MOVAB	32(ORB), LOCAL_ORB+32	0596
2C		AE	24	A6	9E	000C0	MOVAB	36(ORB), LOCAL_ORB+36	0597
20		A6		10	88	000C5	BISB2	#16, 32(ORB)	0598
24		A6		10	88	000C9	BISB2	#16, 36(ORB)	0599
		5B	2A	A1	3C	000CD	MOVZWL	42(R1), SEG_NUMBER	0600
				39	11	000D1	BRB	12\$	0589
005B	8F	00		00	2C	000D3	11\$: MOV C5	#0, (SP), #0, #88, LOCAL_ORB	0604
				08	AE	000DA			

	56	08	AE	9E	000DC	MOVAB	LOCAL ORB, ORB	:	0605
08	A6	58	8F	9B	000E0	MOVZBW	#88, 8(ORB)	:	0606
0A	A6	49	8F	90	000E5	MOVAB	#73, 10(ORB)	:	0607
0B	A6		01	88	000EA	BISB2	#1, 11(ORB)	:	0608
	50	08	AC	D0	000EE	MOVL	HEADER, R0	:	0609
18	A6	0A	A0	B0	000F2	MOVW	10(R0), 24(ORB)	:	
	66	08	A0	9B	000F7	MOVZBW	8(R0), (ORB)	:	0610
02	A6	09	A0	9B	000FB	MOVZBW	9(R0), 2(ORB)	:	0611
	51	01	A0	9A	00100	MOVZBL	1(R0), R1	:	0612
	6E		6041	3E	00104	MOVAB	(R0)[R1], MAP_AREA	:	
	5B	00	BE	9A	00108	MOVZBL	@MAP_AREA, SEG_NUMBER	:	0613
OC	FEE1	CF	04	AC	E1 0010C	BBC	ACCESS, EXT_HEADER, 13\$	:	0619
				0A	15 00113	BLEQ	13\$	:	0622
	05	0000G	CF	E8	00115	BLBS	CLEANUP_FLAGS+1, 13\$	:	
	58		24	D0	0011A	MOVL	#36, STATUS	:	0625
			49	11	0011D	BRB	14\$	:	0626
F4	AD		57	9A	0011F	MOVZBL	FILE_ACCESS_BITS, CHPCTL	:	0633
		04	AE	D4	00123	CLRL	PRIVS_USED	:	0634
	53	60	AE	9E	00126	MOVAB	CHPRET, R3	:	0636
	52	F4	AD	9E	0012A	MOVAB	CHPCTL, R2	:	
	51		56	D0	0012E	MOVL	ORB, R1	:	
	50		5A	D0	00131	MOVL	ARB, R0	:	
		00000000G	00	16	00134	JSB	EXE\$CHKPRO_INT	:	
	58		50	D0	0013A	MOVL	R0, STATUS	:	
	3D		58	E8	0013D	BLBS	STATUS, 17\$	:	0642
	04	04	AC	D1	00140	CMPL	ACCESS, #4	:	0645
			22	12	00144	BNEQ	14\$	:	
F4	AD		01	8A	00146	BICB2	#1, CHPCTL	:	0648
F4	AD		10	88	0014A	BISB2	#16, CHPCTL	:	0649
		04	AE	D4	0014E	CLRL	PRIVS_USED	:	0650
	53	60	AE	9E	00151	MOVAB	CHPRET, R3	:	0651
	52	F4	AD	9E	00155	MOVAB	CHPCTL, R2	:	
	51		56	D0	00159	MOVL	ORB, R1	:	
	50		5A	D0	0015C	MOVL	ARB, R0	:	
		00000000G	00	16	0015F	JSB	EXE\$CHKPRO_INT	:	
	58		50	D0	00165	MOVL	R0, STATUS	:	
	12		58	E8	00168	BLBS	STATUS, 17\$	:	0662
	0B	14	AC	E8	0016B	BLBS	REQUIRED, 16\$	:	0663
	69		57	91	0016F	CMPB	FILE_ACCESS_BITS, (R9)	:	0664
			06	13	00172	BEQL	16\$	:	
	57		69	90	00174	MOVAB	(R9), FILE_ACCESS_BITS	:	0667
		FEAE	31	00177	BRW	1\$		:	0537
	0F		58	E9	0017A	BLBC	STATUS, 18\$	:	0674
		10	AC	D5	0017D	TSTL	ALT_ACCESS	:	0675
			0A	13	00180	BEQL	18\$	:	
	69		57	91	00182	CMPB	FILE_ACCESS_BITS, (R9)	:	0676
			05	12	00185	BNEQ	18\$	:	
	58	0681	8F	3C	00187	MOVZWL	#1665, STATUS	:	0677
	50		58	D0	0018C	MOVL	STATUS, R0	:	0679
			04	0018F	RET			:	0681

; Routine Size: 400 bytes, Routine Base: \$CODE\$ + 0041

: 369 0682 1
: 370 0683 1 END
: 371 0684 0 ELUDOM

PSECT SUMMARY

```
:
:      Name      Bytes      Attributes
: $CODE$      465 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
```

Library Statistics

```
:
:      File      ----- Symbols ----- Pages Processing
:      File      Total   Loaded   Percent   Mapped   Time
: _$255$DUA28:[SYSLIB]LIB.L32;1  18619      41      0      1000      00:01.9
```

COMMAND QUALIFIERS

```
:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:CHKPRO/OBJ=OBJ$:CHKPRO MSRC$:CHKPRO/UPDATE=(ENH$:CHKPRO)
```

```
: Size:      443 code + 22 data bytes
: Run Time:   00:13.8
: Elapsed Time: 00:40.7
: Lines/CPU Min: 2978
: Lexemes/CPU-Min: 15866
: Memory Used: 173 pages
: Compilation Complete
```


0164 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY